REMARKS

Applicants would like to thank the Examiner for granting the interview held on June 14. Claim 18-37 are pending in this application. The Examiner has objected to dependent claims 34-37 for informalities related to the proper notation for adding and deleting text from claims under the revised amendment practice. Applicants have corrected dependent claims 34-37 to conform to the revised amendment practice as suggested by the Examiner. The Examiner has also rejected claims 18-37 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent Application No. US2002/0100029 (*"Bowen"*) filed by Matt Bowen. Applicants have amended claims 18, 20, 21, 26, 28, 33, and 35 to clarify the meaning of "undeclared variable" as discussed below.

Applicants respectfully submit that *Bowen* neither discloses nor suggests the pending claims. According to M.P.E.P §706.02, "for anticipation under 35 U.S.C. 102, the reference must teach *every aspect* of the claimed invention..." (emphasis added). However, *Bowen* fails to disclose or suggest various elements of the claimed invention.

The Official Action states that the parser/parsing step of claims 18, 25, and 26 is disclosed in the *Bowen* reference in Fig. 2 and pages 19-32. More specifically, the Official Action states that "since the term 'undeclared variable' is not clearly defined in the specification of the instant invention, any temporary or unsigned variables used in 'if', 'while' or 'for' statements i.e., **ir** in page 20, line 14 of Bowen read as the 'undeclared variable.'" Official Action, p. 3, 6. This position was confirmed during the Examiner interview.

In order to more provide additional clarity to the term "undeclared variable," Applicants have amended the claims to replace the term, "undeclared variable" with the term, "variable with unknown type or dimension." Applicants submit that this change does not modify the scope of

the claims in any way because assigning a type to a variable is providing a declaration. Conversely, the absence of a type assignment results in an undeclared variable. Thus, changing the claims from "undeclared variable" to "variable with unknown type or dimension" is merely more descriptive of the concept previously claimed and described in the application.

The *Bowen* reference does not disclose the claimed parser for "parsing the functional description expressed in the interpretive, algorithmic language with ***at least one variable of unknown type or dimension*** into an abstract syntax tree." (emphasis added). The claim language specifically recites parsing a functional description from an interpretive language "containing at least one ***variable of unknown type or dimension.***" The use of an input language with variables of unknown type or dimension distinguishes the present invention from *Bowen* and other similar references disclosing the use of the C language.

In contrast to the present invention which accepts as input, languages containing variables of unknown type or dimension (i.e. variables where the type or dimension is ***undeclared***), the *Bowen* reference discloses a method of parsing and compiling only C language programs. In *Bowen,* the language disclosed is ANSI C. *Bowen* ¶ 61. C language programs, by definition, cannot contain any variables of unknown type or dimension. Applicants respectfully submit that ANSI C is a language that is well known in the field to require that the type of all variables is explicitly declared. For example, the seminal work in the field establishing the ANSI C language, "The C Programming Language" by Brian Kernighan and Dennis Ritchie states:

> In C, all variables must be declared before they are used, usually at the beginning of the function before any executable statements. A *declaration* announces the properties of variables; it consists of a type and name and a list of variables such as
>
> ```
> int fahr, celcius;
> int  lower, upper, step;
> ```
>
> The type `int` means that the variables listed are integers, by contrast with `float` which means floating point, i.e. numbers that have a fractional part. The range of both `int` and

> `float` dependes on the machine you are using. 16-bit ints, which lie between -32768 and +32767 are common, as are 32 bit ints. A `float` is typically a 32-bit quantity, with at least six significant digits and magnitude generally between about 10^{-38} to 10^{38}. *Kernighan and Ritchie*, p. 9 (emphasis original).

Thus, it is clear that the C language, and specifically the use of that language in the *Bowen* reference, requires the declaration of types of all variables.

The Offical Action references source code from Appendix 1 of the *Bowen* reference to support the position that "any temporary or unsigned variables used in 'if', 'while' or 'for' statements i.e., **ir** in page 20, line 14 of Bowen read as the 'undeclared variable.'" Official Action, p. 3, 6. In this source code example, **ir** is explicitly declared as an unsigned integer, as required by the C language, on page 19, line 12 in the following statement:

> ram memory[1<<mw] unsigned iw PC, ir, tos;

This declaration is specifically explained in the *Bowen* specification:

> In this embodiment the user writes a description 202 of the system in a C-like language, which is actually ANSI C with some additions which allow efficient translation to hardware and parallel processes. This input description will be compiled by the system 200 of FIG. 2. The additions to the ANSI C language include the following: ***Variables are declared with explicit bit widths and the operators working on the variables work with an arbitrary precision.*** This allows efficient implementation in hardware. For instance a statement which declares the width of variables (in this case the program counter pc, the instruction register ir, and the top of stack tos) is as follows:
>
> unsigned 12 pc, ir, tos.

Bowen, ¶¶ 61-63 (emphasis added). This description clearly demonstrates that *Bowen* parser requires variable types to be ***declared***. Moreover, nowhere does *Bowen* describe the parsing or processing of ***variables with unknown type or dimension.*** Therefore, the parser/parsing step for parsing ***at least one variable of unknown type or dimension*** of claims 18, 25 and 26 is not disclosed by *Bowen.*

Additionally, the *Bowen* reference specifically states that the type variable **ir** is ***declared*** as an unsigned interger in the source code in Appendix 1:

> Referring to Appendix 1, the processor starts with a definition of the instruction width, and the width of the internal memory and stack addresses. This is followed by an assignment of the processor opcodes. Next the registers are defined; ***the declaration "unsigned x y, z" declares unsigned integers y and z of width x.*** The program counter, instruction register and top-of-stack are the instruction width; the stack pointer is the width of the stack.

Bowen ¶ 173. Thus, there is simply no question that the variable **ir** in Appendix 1 is specifically a variable with a known type.

During the interview, the Examiner stated that even though the line "unsigned iw PC, ir, tos" appears in the source code of Appendix 1, the term "unsigned" somehow reads on "undeclared." However, as demonstrated above, "unsigned" does not mean "undeclared" in *Bowen* or the C language. In the C language, "unsigned" specifically refers to a type of variable. For example, in a given system that uses 16 bits to represent an `int`, a variable declared as `int` would have a numerical range from -32768 and +32767, whereas a variable declared as `unsigned` has a range of 0 to +65556. "Unsigned" is indeed part of the statement declaring a data type for the variables PC, ir, and tos, as described in *Bowen* at ¶ 173.

Finally, the Office Action states that *Bowen* reads upon "undeclared variable" because "undeclared variable" is not clearly defined in the application for the present invention. Applicants respectfully disagree. For example, the application for the present invention specifically treats the concept of declaring the types of variables and distinguishes that practice from the present invention:

> In particular, during operation of present methodology, given certain types and shapes of variables, for example, C-code may be ***generated to declare*** variables and corresponding statements...
>
> Following is a sample MATLAB code:
> *a = b + 2;*

where the correspondingly generated C code is:

float a[100; 200]; b[100; 200] ;

App. p. 12 (emphasis added). Moreover, the application explains in several places that interpretive languages do not use variables of known types and that the present invention must determine and provide declarations for the variables:

- "MATLAB variables have no notion of type or shape...." App. p. 8.

- "Hence, to compile and synthesize program written in MATLAB, such that maximum information about type and shape of arrays in particular, and of variables in general, are determined appropriately, algebraic framework is thereby provided to determine type and shape of arrays preferably at compile time." App. p. 10.
- "Thus, to scalarize MATLAB vector constructs, array shape and size are determined; although MATLAB is dynamically typed and may not ordinarily provide explicit basic data type and shape declarations. Accordingly, in accordance with one aspect of present invention, type-shape analysis 14 is applied." App. p. 12.
- "Present compiler declares scalars as variables to facilitate movement of operations across states by optimization phases. Variables corresponding to function arguments are declared signals to be visible outside the process corresponding to the function. Other signal declarations include signals corresponding to memory interface." App. p. 16.

To the extent that these references do not clearly define "undeclared" applicants respectfully suggest that this term is well known to those of ordinary skill in the art as demonstrated abover through references to *Bowen* and the C language.

The claims are also distinguished from *Bowen* and the C language in other ways. For example, the claims recite "compiling a functional description expressed in an ***interpretive, algorithmic language***," and "parsing the functional description expressed in the ***interpretive, algorithmic language***." A compiled language is defined as "a language that is translated into machine code prior to any execution as opposed to an interpreted language, which is translated and executed statement by statement." Microsoft Computer Dictionary, p. 100 (Microsoft Press, 1999).

"C is a compiled language...." *Id.* at 69. Because C is a compliled language, it cannot be an interpretive language as recite by the pending claims. Therefore, the claims are distinguished from *Bowen* and the C language.

Conclusion

In sum, Applicants respectfully submit that claims 18-37 as presented herein, are patentably distinguishable over the cited references (including references cited, but not applied). Therefore, Applicants request reconsideration and allowance of these claims.

Applicants respectfully invite Examiner to contact Applicants' representative at the number provided below if Examiner believes it will help expedite furtherance of this application.

RESPECTFULLY SUBMITTED,
Prithviraj Banerjee, Alok Choudhary, Malay Haldar, Anshuman Nayak

Date: 07/20/05

By: Deepti Pan

Deepti Panchawagh-Jain, Esq.
Registration No. 43,846
3039 Calle De Las Estrella
San Jose, CA 95148
(408) 274-3043
(408) 516-5825